Programmeringsteknik för Bio1 och I1
Övning 4

Övningsgrupp 3 (Sal E33)

Johannes Hjorth
hjorth@nada.kth.se
Rum 4538 på plan 5 i D-huset
08 - 790 69 02

Kurshemsida:
http://www.nada.kth.se/kurser/kth/2D1310/

Övningsanteckningar och diagnostiska prov:
http://www.nada.kth.se/∼hjorth/teaching/

Administrativt

Kontrollera att ni har fått Lab2 och Lab3 inrapporterad i res-systemet genom att skriva

```
>res show prgibio04
```

```
RAPPORTERADE RESULTAT FÖR: ÅÅMMDD-XXXX Efternamn, Förnamn
ANVNAMN: användarnamn STUDIESTATUS: I-03 GRUPP: 3
Moment   Nr    Datum      Resultat   Rapp av

lab      1     040123     G          auto
lab      2     040204     G          hjorth
lab      3     040211     G          hjorth
```

Säg till om något är fel!

Kom ihåg de diagnostiska proven! Frågorna som kommer på tentan kommer vara snarlika, men inte exakt likadana.

Jag länkar till proven från min websida

`http://www.nada.kth.se/∼hjorth/teaching`

Dagens program

Idag ska vi lära oss hantera stora mängder data.

Hur gör vi om vi vill lagra 100 olika värden, har vi då hundra olika variabler eller finns det ett bättre sätt?

Vi kommer titta på arrayer och klassen `Vector`.

Vad är skillnaden mellan de två?

Vilken ska vi välja?

Hur använder vi dem?

Vi kommer se hur man kan återanvända kod.

Mindre kod att skriva, färre ställen att göra fel på, det går alltså fortare att skriva programmen.

Introduktion till arrayer

Med arrayer kan vi använda samma variabelnamn till att lagra flera olika värden.

När vi arbetar med arrayer kommer vi ofta också att använda oss av `for`-loopar.

Arrayer och `for`-loopar hör ihop!

För att skapa en variabel brukar vi skriva,

```
int x = 0;
```

För att istället skapa en array skriver vi,

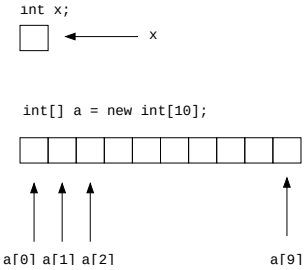
```
int[] a = new int[10];
```

Hur ska raden tolkas?

Hakparenteserna efter `int` säger att det är en array och med `new` skapar vi här plats för 10 heltal.

Hur når vi de olika elementen?

Variabeln `x` innehåller bara ett värde, men hur gör vi om det ligger 10 olika värden i arrayen `a`?



Vi numrerar positionerna för att skilja dem åt!

Första elementet, som har index noll, når man med

```
int x = a[0];
```

Här har vi angivit indexet innanför hakparenteserna.

Arrayer och `for`-loopar

Arrayer kombineras med fördel med `for`-loopar

Om vi vill sätta alla element i `a` till 17 skriver vi

```
for(int i = 0; i < a.length; i++) {
    a[i] = 17;
}
```

Eftersom `a.length` är lika med antalet element i arrayen `a` kommer vi att successivt öka `i` tills vi har stegat igenom alla element.

Array med `Jacuzzi`

Det går även att skapa arrayer fyllda med objekt!

Vi skapar en array med `Jacuzzi`-objekt så här

```
Jacuzzi[] lokaltParadis = new Jacuzzi[5];
```

Med en `for`-loop instansierar vi sedan alla objekt

```
for(int i = 0; i < lokaltParadis.length; i++) {
    lokaltParadis[i] = new Jacuzzi(30);
}
```

För att ändra temperaturen på det *fjärde* elementet anropar vi instansmetoden `setTemperatur`

```
lokaltParadis[3].setTemperatur(41);
```

Notera att fjärde elementet har index *tre*!

`Jacuzzi`-array exempel

Exempel på en array med `Jacuzzi`-objekt

```
import java.io.*;

public class FleraJacuzzi {

    public static void main(String[] args) {
        int i; // loopvariabel

        // Vi skapar en array med plats för fem Jacuzzi
        Jacuzzi[] lokaltParadis = new Jacuzzi[5];

        // Vi instansierar de fem Jacuzzi objekten
        for(i = 0; i < lokaltParadis.length; i++) {
            lokaltParadis[i] = new Jacuzzi(30);
        }

        // Vi ändrar temperaturen på det fjärde elementet
        lokaltParadis[3].setTemperatur(41);

        for(i = 0; i < lokaltParadis.length; i++) {
            System.out.println(lokaltParadis[i]);
        }
    }
}
```

Vi kompilerar och kör `FleraJacuzzi.java`

```
>javac FleraJacuzzi.java
>java FleraJacuzzi
Jacuzzin har temperatur 30 grader
Jacuzzin har temperatur 30 grader
Jacuzzin har temperatur 30 grader
Jacuzzin har temperatur 41 grader
Jacuzzin har temperatur 30 grader
```

Arrayer räcker inte alltid...

När vi använder arrayer måste vi på förhand bestämma hur många element som ska få plats.

```
Jacuzzi[] lokaltParadis = new Jacuzzi[5];
```

Vad händer om vi plötsligt behöver lägga till ytterligare ett element? Problem!

```
lokaltParadis[5] = new Jacuzzi();
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at FleraJacuzzi.main(Compiled Code)
```

Vore det inte smidigt om vi slapp oroa oss för hur många element som fick plats?

Det finns ett sätt...

Vi använder klassen `Vector` istället!

Klassen `Vector` kan öka sin storlek vid behov!

För att kunna använda `Vector` behöver vi skriva

```
import java.util.*;
```

Vi skapar en `Vector` genom att skriva,

```
Vector jacuzzis = new Vector();
```

För att lägga till ett element skriver vi

```
jacuzzis.addElement(new Jacuzzi(40));
```

För att titta på *tredje* elementet skriver vi

```
Jacuzzi temp = (Jacuzzi) jacuzzis.elementAt(2);
```

Eftersom `Vector` kan innehålla objekt av olika typ måste vi berätta att det är en `Jacuzzi` vi hämtar, vilket vi gör genom att skriva `(Jacuzzi)`.

Anropa instansmetoder!

Hur anropar vi de olika elementens instansmetoder?

Först hämtar vi ut objektet ur vektorn `jacuzzis`

```
Jacuzzi kall = (Jacuzzi) jacuzzis.elementAt(0);
```

Sedan kan vi anropa objektets instansmetod

```
kall.setTemperatur(-273);
```

Eftersom vi arbetar med referenser så ändras temperaturen på den första Jacuzzin i vektorn.

Vi behöver alltså inte stoppa tillbaka `kall` i `jacuzzis` för att första elementet ska ändras!

Java API är din vän!

Klassen `Vector` har många olika metoder för att sätta in, ta bort och ändra element.

Hos Java API finner ni alla olika `Vector`-metoder

`http://www.sgr.nada.kth.se/unix/docs/java/api/`

Några exempel,

`int indexOf(Object elem)`
Searches for the first occurence of the given argument, testing for equality using the equals method.

`void setElementAt(Object obj, int index)`
Sets the component at the specified index of this vector to be the specified object.

`int size()`
Returns the number of components in this vector.

Ett `Vector` exempel

Ett exempel på hur man kan skapa en vektor med `Jacuzzi`-objekt och sedan anropar en instansmetod för att ändra ett av elementens temperatur.

```
import java.io.*;
import java.util.*;

public class VectorJacuzzi {

    public static void main(String[] inargument) {
        Jacuzzi tmp; // Temporärarvariabel

        Vector jacuzzis = new Vector();

        jacuzzis.addElement(new Jacuzzi(40));
        jacuzzis.addElement(new Jacuzzi(43));
        jacuzzis.addElement(new Jacuzzi(37));

        System.out.println("Antal element i vektorn: " + jacuzzis.size());

        for(int i = 0; i < jacuzzis.size(); i++) {
            tmp = (Jacuzzi) jacuzzis.elementAt(i);
            System.out.println("Element " + i + ": " + tmp);
        }

        System.out.println("*Vi ändrar på temperaturen på första elementet");
        tmp = (Jacuzzi) jacuzzis.elementAt(0); // Hämta referesen
        tmp.setTemperatur(-273);               // Ändra temperaturen

        for(int i = 0; i < jacuzzis.size(); i++) {
            tmp = (Jacuzzi) jacuzzis.elementAt(i);
            System.out.println("Element " + i + ": " + tmp);
        }
    }
}
```

Kort om `VectorJacuzzi.java`

Vi kompilerar och kör koden, inga överaskningar här

```
>javac VectorJacuzzi.java
>java VectorJacuzzi
Antal element i vektorn: 3
Element 0: Jacuzzin har temperatur 40 grader
Element 1: Jacuzzin har temperatur 43 grader
Element 2: Jacuzzin har temperatur 37 grader
*Vi ändrar på temperaturen på första elementet
Element 0: Jacuzzin har temperatur -273 grader
Element 1: Jacuzzin har temperatur 43 grader
Element 2: Jacuzzin har temperatur 37 grader
```

Genom att använda `jacuzzis.size()` i våra `for`-loopar kan vi enkelt stega igenom alla element.

Hade vi lagt till ett objekt till i vektorn hade looparna fortfarande fungerat.

Om ni undrar över vilka andra metoder som `Vector` eller någon annan klass har, titta då i Java API.

Lab5 – Vad ska vi göra?

Det är meningen att ni utan att ändra något ska kunna utnyttja er gamla `Bil`-klass från Lab4.

Skapa `ArrayBil.java` vilken innehåller en `main`-metod samt en array med `Bil`-objekt.

När ni fått det att fungera skapa då `VectorBil.java` som också kommer ha en `main`-metod samt en `Vector` med `Bil`-objekt.

Ni ska använda `for`-loopar för att stega igenom arrayen respektive vektorn och anropa varje elements instansmetod(er) för att beräkna årlig körsträcka samt kilometerkostnad.

Båda dessa program `ArrayBil` och `VectorBil` kommer utnyttja `Bil.java`.

Detta är ett exempel på att återanvända kod!